Simulación de plasma electrostático usando el paquete de librerías ANACONDA mediante el método Particle-In-Cell

Juan Sebastián Blandón Luengas
Juan Pablo Grisales Campeón

Universidad Tecnológica de Pereira
Facultad de Ingenierías
Departamento de Ingeniería Física
Pereira
2016

Simulación de plasma electrostático usando el paquete de librerías ANACONDA mediante el método Particle-In-Cell

Juan Sebastián Blandón Luengas
Juan Pablo Grisales Campeón

Tesis de grado para optar al título de:
Ingeniero Físico

Director
Ph.D. Henry Riascos Landázuri

Universidad Tecnológica de Pereira
Facultad de Ingenierías
Departamento de Ingeniería Física
Pereira
2016

Índice general

Índice general I

Índice de figuras IV

Agradecimientos VI

Abstract VII

Resumen VIII

I Preliminares 1

1. Introducción 2

2. Objetivos 4

II Marco teórico 5

3. Parámetros del plasma 6
3.1. Longitud de Debye λ_d 6
3.2. Frecuencia del plasma ω_p 6

4. Modelo teórico 8
4.1. Caso electrostático 8
4.2. Oscilaciones de plasma frío 8
4.3. Inestabilidad Two-Stream 9
4.4. Inestabilidad Beam-Plasma 10
4.5. Amortiguamiento de Landau 10
4.6. Campos magnéticos aplicados 11

5. Modelo computacional 13
5.1. Discretización de las ecuaciones de movimiento y campo 13
5.1.1. Integrador *Leap-Frog* 13
5.1.2. Método del trapecio 14
5.1.3. Método de Boris 14

5.2. Normalización . . . 15
5.3. Condiciones de frontera . . . 16
5.4. Método Particle-In-Cell . . . 16
5.4.1. Ciclo computacional . . . 16
5.4.2. Configuración inicial . . . 17
5.4.3. Malla euleriana . . . 18
5.4.4. Malla lagrangiana . . . 18
5.4.5. Métodos de ponderación . . . 18

III Implementación y simulación 20

6. Metodología 21

7. Desarrollo del PC-GPLA.py 23
7.1. Subrutina `cargarposicion` . . . 24
7.2. Subrutina `cargarvelocidad` . . . 24
7.3. Subrutina `particula_cf` . . . 24
7.4. Subrutina `densidad` . . . 25
7.5. Subrutina `campo` . . . 25
7.6. Subrutina `movimientoparticulas` . . . 25
7.7. Subrutina `diagnosticos` . . . 25
7.8. Subrutina `historial` . . . 25
7.9. Subrutina `relacion_dispersion` . . . 26
7.10. Subrutina `gestion_archivos` . . . 26
7.11. Subrutina `nextpow2` . . . 26
7.12. Programa principal . . . 27

IV Discusión y resultados 28

8. Discusión y resultados 29
8.1. Oscilaciones de plasma frío . . . 29
8.2. Inestabilidad *Two-Stream* . . . 30
8.3. Inestabilidad *Beam-Plasma* . . . 32
8.4. Amortiguamiento de Landau . . . 33
8.5. Campo magnético uniforme aplicado a un flujo de electrones . . . 34

V Conclusiones y Trabajo Futuro 36

9. Conclusiones y trabajo futuro 37
9.1. Conclusiones . . . 37
9.2. Artículos publicados . . . 37
9.3. Trabajo futuro . . . 38

VI Anexos **39**

A. Código Plasma Computacional Grupo Plasma Láser y Aplicaciones **40**

B. Transformada Rápida de Fourier **41**

C. Relaciones de dispersión **42**
C.1. Relación de dispersión de oscilaciones de plasma frío 42
C.2. Relación de dispersión ITS 43

Bibliografía **45**

Índice de figuras

4.1. Oscilación de la energía cinética obtenida por Martin [1]. 9
4.2. Esquema de las funciones de distribución de la ITS [2]. 9
4.3. Esquema de las funciones de distribución de la IBP. 10
4.4. Comparación de la velocidad de las partículas respecto a v_{ph} a través de la función de distribución [3]. 11
4.5. Trayectoria de una partícula al estar afectada por un campo eléctrico y un campo magnético uniforme. 11

5.1. Esquema de integración del método LF [4]. 13
5.2. Esquema gráfico del método del trapecio acumulativo [5]. 14
5.3. Diagrama de flujo del método PIC. 17
5.4. Forma efectiva de las partículas para el *Nearest Grid Point* (NGP) o esquema de orden 0, y para el *Cloud-In-Cell* (CIC) o esquema de orden 1. . . . 19

8.1. Campo eléctrico en $t = 150$ para oscilaciones de plasma frío. 29
8.2. Energías para oscilaciones de plasma frío. 29
8.3. Relación de dispersión para oscilaciones de plasma frío: teórica (línea negra constante) y simulada(diagrama de colores). 30
8.4. Espacio de fase para ITS en plasma frío: sin perturbación. 31
8.5. Espacio de fase para ITS en plasma frío: amplitud de la perturbación de 0,1. 31
8.6. Espacio de fase para ITS en plasma frío: amplitud de la perturbación de 0,5. 32
8.7. Relación de dispersión para la ITS en plasma frío: teórica y simulada sin perturbación. 32
8.8. Relación de dispersión para la ITS en plasma frío: teórica y simulada con perturbación de 0,1 de amplitud. 32
8.9. Relación de dispersión para la ITS en plasma frío: teórica y simulada con perturbación de 0,5 de amplitud. 33
8.10. Espacio de fase para IBP en plasma frío: sin perturbación. 33
8.11. Amortiguamiento de Landau en el espacio de fase para la ITS 34
8.12. Amortiguamiento de Landau en el espacio de fase para la IBP. 34
8.13. Amortiguamiento de Landau visto a través de la energía del campo eléctrico para la ITS 34
8.14. Amortiguamiento de Landau visto a través de la energía del campo eléctrico para la IBP. 34
8.15. Trayectoria de SPs afectadas por un campo magnético externo uniforme ($Bo = 5$) 35

8.16. Trayectorias SPs distintas bajo la acción de campos eléctrico y magnético ($Bo = 5$) perpendiculares entre sí. 35

A.1. Diagrama de flujo del `PC-GPLA.py`. 40

B.1. Esquema del cálculo del algoritmo de Cooley-Tukey o FFT. 41

C.1. Relación de dispersión para el plasma frío. 43
C.2. Relación de dispersión para la ITS considerando uno solo de los cuadrantes [2]. 44

Agradecimientos

Agradecemos al profesor Henry Riascos Landázuri y al Grupo Plasma, Láser y Aplicaciones (GPLA) por su apoyo durante el desarrollo de éste trabajo de grado.

También agradecemos a la Universidad Tecnológica de Pereira por abrirnos las puertas a un mar de infinito conocimiento.

Asimismo, agradecemos al Servicio Nacional de Aprendizaje (SENA) y a Colciencias por la financiación brindada a través de la convocatoria "Jóvenes Investigadores e Innovadores 2015".

Finalmente, agradecemos a nuestra familias, amigos y conocidos, que de una u otra manera apoyaron este camino emprendido años atrás y que hoy muestra los frutos del trabajo arduo y constante.

Y a quienes no hayamos nombrado explícitamente: gracias totales.

Abstract

In this work was build a Python code using ANACONDA packages, which simulates electrostatic and electromagnetic plasma through various phenomena. It was used a specialized algorithm to simulate many particles system dynamics called Particle-In-Cell (PIC). Computational plasma was corroborated simulating three phenomena which can be studied using electrostatic and electromagnetic models: cold plasma oscillations, Two-Stream and Beam-Plasma instabilities. Besides, an external uniform magnetic field was added to code to achieve an electromagnetic behaviour. Then, this code allows to know and to study numerical methods, which simulates plasma dynamics under approximations and understand its physical mechanisms. So, its importance lies in computational physics line creation at Grupo Plasma, Láser y Aplicaciones.

Resumen

En este trabajo se construyó un código en Python con el paquete de librerías ANACONDA, que simula plasma electrostático y electromagnético a través de distintos fenómenos. Se usó un algoritmo especializado para simular la dinámica de sistemas de muchas partículas denominado Particle-In-Cell (PIC). Para corroborar la simulación de plasma se simularon tres fenómenos que siguen el modelo electrostático y unidimensional, los cuales fueron: oscilaciones de plasma frío, y las inestabilidades Two-Stream (ITS) y Beam-Plasma (IBP). Además, se añadió la posibilidad de agregar un campo magnético externo uniforme para obtener un comportamiento electromagnético. Así, el código aquí implementado es una herramienta que permite conocer y estudiar los métodos numéricos para simular la dinámica del plasma bajo la aproximación usada y entender los mecanismos físicos detrás de ella, aspecto de gran importancia para la creación de la línea de física computacional en el Grupo Plasma, Láser y Aplicaciones.

Parte I

Preliminares

Capítulo 1

Introducción

La física del plasma es una disciplina de gran importancia en la actualidad, dado que este forma alrededor del 99 % de la materia presente en el universo que habitamos [6]. Sin ir muy lejos, campos como la fusión nuclear, las telecomunicaciones, la nanotecnología, entre otros, requieren de desarrollos acertados tanto teóricos como experimentales, para explicar los mecanismos físicos de fenómenos que no son totalmente entendidos en este tipo de entidades colectivas [6]. En este trabajo se estudia el caso electrostático, que es importante puesto que su comportamiento es ideal y representativo, ya que permite un entendimiento parcial de fenómenos como oscilaciones, inestabilidades y amortiguamiento. Esto crea el marco sobre el que se pueden estudiar casos cercanos a la realidad.

La simulación de plasma ha cumplido un papel importante en el desarrollo de la física detrás de este, ya que es una herramienta útil para la validación de modelos teóricos, que de ser llevados a la práctica, exigirían el desarrollo de nuevas tecnologías que en la mayoría de los casos son costosas, o incluso, no han sido creadas. Este interés por desarrollar simulaciones en ésta área surgió durante el siglo anterior, cuando Birdsall y Langdon publicaron por primera vez en 1985 su libro *Plasma physics via computer simulation* [2]; el material provee las bases para la construcción de códigos que permitan la representación computacional de diversos fenómenos presentes en el plasma tales como: oscilaciones, ondas, inestabilidades y amortiguamiento. Un ejemplo de aplicación es la simulación de la ITS en plasma electrostático, que ya había sido tratada con anterioridad por Morse y Nielson , quienes entregaron resultados para una, dos y tres dimensiones [7]. Sin embargo, las herramientas que han surgido desde entonces han permitido que se desarrollen trabajos más precisos, que posibilitan la explotación de los recursos computacionales disponibles, como el desarrollado por Martin quien construyó un código en C, que simula oscilaciones e inestabilidades *Two-Stream* y *Four-Stream* en plasma electrostático y frío [1]; o la simulación de IBP llevada a cabo por Fox, Williams y Messina [26] mediante un código paralelizado. Teniendo en cuenta lo anterior, se agrega que el Grupo Plasma, Láser y Aplicaciones (GPLA) no ha sido ajeno a la simulación, tal como lo evidencia el trabajo desarrollado por González [8], quien simuló la dinámica de expansión de una pluma de plasma generado por ablasión láser mediante el método *Particle-In-Cell*.

De acuerdo a lo expuesto previamente, el trabajo aquí presentado es un código desarrollado en Python que modela plasma en una dimensión, con una extensión a 1.5D, usando el paquete de librerías ANACONDA [9], en el que se implementó el método computacio-

nal *Particle-In-Cell*(PIC), que fue corroborado mediante el análisis de cuatro fenómenos característicos del plasma electrostático y frío: inestabilidades *Two-Stream* (ITS) y *Beam-Plasma* (IBP), oscilaciones y amortiguamiento de Landau (AL); además se incluyó la posibilidad de afectar los sistemas con un campo magnético externo. Lo anterior se alcanzó con gráficas en el espacio de fase para observar el comportamiento temporal, y con las respectivas relaciones de dispersión para comparar los datos de la simulación con los esperados en términos del campo simulado. De ésta manera, la importancia de ésta investigación radica en el hecho de que a partir de ésta simulación se logrará un primer acercamiento a comportamientos complejos que sentarán las bases para modelamientos más elaborados, así como, la creación de una nueva línea de investigación en el Grupo Plasma, Láser y Aplicaciones (GPLA) dedicada al estudio computacional del plasma (Departamento de Simulación).

Capítulo 2

Objetivos

Objetivo general

Construir un código mediante el algoritmo Particle-In-Cell (PIC) usando el paquete de librerías ANACONDA para simular fenómenos en plasma unidimensional y sentar las bases para la creación de una nueva rama de investigación en el Grupo Plasma, Láser y Aplicaciones.

Objetivos específicos

1. Implementar computacionalmente los métodos numéricos necesarios para el algoritmo PIC.
2. Identificar fenómenos característicos y representativos del modelo de plasma electrostático y simularlos con PIC
3. Implementar técnicas que permitan excitaciones externas al sistema tales como: perturbaciones y campos.
4. Construir una versión final del código que abarque todos los fenómenos simulados.
5. Presentar el trabajo a la comunidad universitaria local y nacional con el fin de dar a conocer el inicio de la rama en estudio computacional del plasma en el GPLA.

Parte II

Marco teórico

Capítulo 3

Parámetros del plasma

El plasma es una entidad cuyo comportamiento colectivo, al estar sujeto a gran cantidad de variables, tales como temperatura, densidad, presión, entre otras, hacen complejo su análisis si de una descripción precisa se refiere. Más sin embargo, se recurren a suposiciones, que al ser comprobadas experimentalmente son relevantes a la hora de estudiar los fenómenos presentes en él. Es aquí donde es importante definir constantes características en el plasma, que están supeditadas, por ejemplo, al número de partículas o a las especies involucradas. Por lo tanto, es pertinente determinar qué es la longitud de Debye (λ_D) [10] y la frecuencia del plasma (ω_p).

3.1. Longitud de Debye λ_d

Es bien sabido que un plasma está compuesto de iones y de electrones. Ahora bien, si se ubica un ión en medio de los electrones, éstos tenderán a crear una especie de coraza alrededor de éste, impidiendo que el campo eléctrico generado afecte al plasma [11], en otras palabras, la carga quedará neutralizada. Dicha coraza se conoce como la esfera de Debye, y el radio de ésta se conoce como radio de Debye o *longitud de Debye* λ_D. La ecuación 3.1 muestra cómo está definida ésta cantidad:

$$\lambda_D = \sqrt{\frac{\epsilon_0 k T_e}{n_e e^2}} \tag{3.1}$$

Donde ϵ_0 es la permitividad eléctrica en el vacío, k es la constante de Boltzmann, T_e, n_e y e, son la temperatura, la densidad y la carga del electrón respectivamente. La importancia de este parámetro radica en el hecho de que para longitudes mayores a la longitud de Debye el sistema presenta comportamiento colectivo y un estado cuasi neutral, las cuáles son características que definen a un plasma.

3.2. Frecuencia del plasma ω_p

De la misma manera que la longitud de Debye proporciona información sobre las escalas de longitud de un plasma, la frecuencia del plasma determina la escala de tiempo de este

y se presenta debido a la interacción electromagnética entre las partículas; está definida como:

$$\omega_p = \sqrt{\frac{n_\alpha e^2}{m_\alpha \epsilon_0}} \tag{3.2}$$

Donde α representa las especies presentes en el plasma y m es la masa.

Capítulo 4

Modelo teórico

En este capítulo se tratarán los temas concernientes a la física que existe detrás de los distintos fenómenos que se desean implementar para su simulación computacional.

4.1. Caso electrostático

Un plasma, se caracteriza por la presencia de campos eléctricos autogenerados, debidos al movimiento de las partículas que conforman el sistema. Ahora bien, un plasma se define como electrostático cuando el campo magnético está ausente ($\vec{B} = 0$). Según lo anterior, el tratamiento que se le puede dar al plasma desde el electromagnetismo comprende la obtención del campo eléctrico a partir de la *ley de Gauss*:

$$\nabla \cdot \vec{E} = \frac{\rho}{\epsilon_0} \tag{4.1}$$

En donde $\vec{E}$ es el campo y ρ la densidad de partículas. La cantidad previa, que se obtiene de acuerdo a las posiciones de las cargas dentro del sistema, permite obtener el campo eléctrico, éste acelera las partículas, tal como lo presenta la *fuerza de Lorentz*:

$$\frac{d\vec{v_p}}{dt} = \frac{q\vec{E_p}}{m_e} \tag{4.2}$$

Donde $d\vec{v_p}/dt$ y $\vec{E_p}$ son la aceleración y el campo asignado a cada partícula.

4.2. Oscilaciones de plasma frío

Considérese una situación en la que el plasma está compuesto por electrones y iones. Ahora, se supone que los iones están en un fondo inmóvil, y para que sea plasma frío, sus velocidades térmicas ($v_{th} = 0$) e iniciales deben ser cero. Si se llega a perturbar el sistema sinusoidalmente, por ejemplo, a través de la posición de los electrones, éstos oscilarán respecto a sus posiciones de equilibrio [6], describiendo así un movimiento armónico simple [2]. Por la razón anterior es que se ha considerado pertinente abordar éste fenómeno, ya que debido a la simpleza de su mecanismo físico, éste representa un buen diagnóstico inicial para el código y el modelo electrostático. En la figura 4.1 se presenta la gráfica de energía

cinética para oscilaciones de plasma frío obtenida por Martin [1], que confirma lo expuesto previamente.

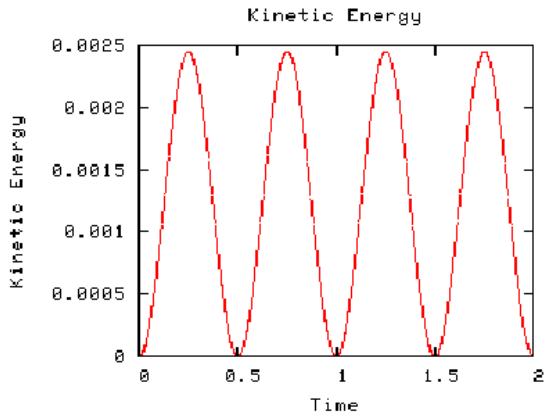


Figura 4.1: Oscilación de la energía cinética obtenida por Martin [1].

4.3. Inestabilidad Two-Stream

La ITS es otro fenómeno bien conocido en plasmas. Este consiste en dos flujos de partículas que se mueven el uno contra el otro [12], generando vórtices entre los haces de éstas que pueden ser observadas a través del espacio de fase. En un sentido más amplio, cuando se habla de ésta inestabilidad en plasmas, la energía cinética de los haces es convertida en energía del campo eléctrico [3].

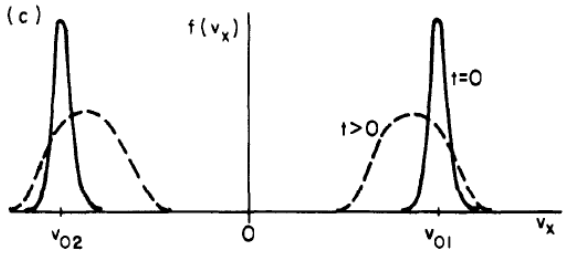


Figura 4.2: Esquema de las funciones de distribución de la ITS [2].

El mecanismo detrás de este tipo de inestabilidad, teniendo en cuenta que se está hablando de plasma, está determinado por una función de distribución Maxwelliana, que modela los dos haces, que en su caso más simple, son de electrones tal como lo muestra la figura 4.2. Los haces tienen una velocidad media igual pero con sentido contrario ($v_{01} = -v_{02}$). En la ecuación 4.3 se presenta la función de distribución que modela el fenómeno:

$$f(x, v_x) = \frac{n_0}{2}\left(\frac{1}{\sqrt{2\pi}v_{th}}e^{\frac{-(v-v_h)^2}{2v_{th}^2}} + \frac{1}{\sqrt{2\pi}v_{th}}e^{\frac{-(v+v_h)^2}{2v_{th}^2}}\right) \tag{4.3}$$

Donde n_0 es la densidad de partículas, v_{th} la velocidad térmica, y v_h la velocidad media del haz. El movimiento de ambos flujos se debe únicamente al campo eléctrico generado por las partículas, así, ésta inestabilidad sigue el modelo electrostático.

4.4. Inestabilidad Beam-Plasma

Este modelo consiste en la inyección de un haz dentro de un plasma estacionario. Para el caso aquí estudiado, se va a considerar un sistema de electrón-electrón con un fondo de iones fijos, tal como se presentó en la sección 4.3. La IBP, a diferencia de la ITS, se genera por una diferencia entre las densidades de los flujos. En el caso acá estudiado, un flujo tendrá el 10 % de la densidad total del plasma (n_{p0}). Es uno de los fenómenos que más se trata en lo que plasma computacional se refiere, puesto que presenta efectos lineales, cuasi-lineales y no-lineales [2]. Hay varias maneras de generar este tipo de inestabilidad, sin embargo, para efectos del trabajo aquí presentado se considera la siguiente: las partículas siguen una distribución de Maxwell con flujos de la misma especie, pero en distintas proporciones cada uno y con velocidades distintas. Uno de los flujos tendrá una v_h cercana a cero:

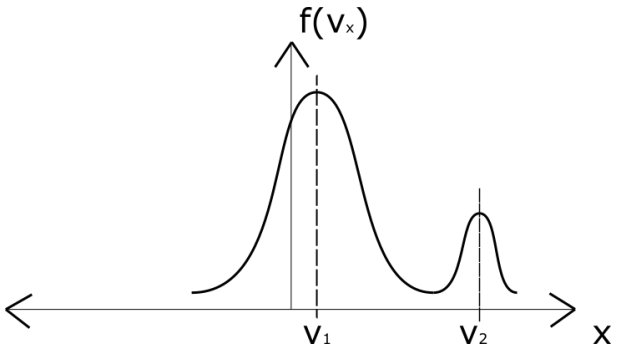


Figura 4.3: Esquema de las funciones de distribución de la IBP.

La ecuación 4.4 muestra la función de distribución que se encarga de describir el fenómeno descrito previamente:

$$f(x, v_x) = \frac{1 - \frac{n_b}{n_0}}{\pi v_{tb}^2} exp\left(-\frac{(v - v_0)^2}{v_{t0}^2}\right) + \frac{\frac{n_b}{n_0}}{\pi v_{tb}^2} exp\left(-\frac{(v - v_b)^2}{v_{tb}^2}\right) \tag{4.4}$$

Donde v_0, v_{t0} y n_0 son velocidad, velocidad térmica y densidad para los electrones cerca al reposo, y v_b, v_{tb} y n_b son las mismas cantidades para los electrones en movimiento [13]. Al igual que en los fenómenos anteriores, el movimiento de las partículas es debido únicamente al campo eléctrico.

4.5. Amortiguamiento de Landau

El AL [14] se debe al intercambio de energía entre una onda y partículas presentes en el plasma con velocidad aproximadamente igual a la velocidad de fase (v_{ph}). Entonces,

dependiendo de las velocidades de las partículas que componen el sistema, éstas se verán afectadas por la onda que describe al campo eléctrico. Así, aquellas partículas con velocidades ligeramente menores que v_{ph} se acelerarán debido al campo, mientras que, las que tengan velocidades ligeramente mayores que v_{ph} se desacelerarán, lo que indica que habrán entregado energía a la onda [3]. La figura 4.4 muestra una comparación gráfica para visualizar qué tan mayor o menor deben ser las velocidades de las partículas respecto al v_{ph}:

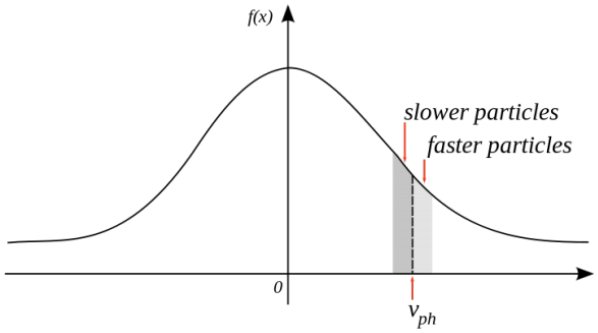


Figura 4.4: Comparación de la velocidad de las partículas respecto a v_{ph} a través de la función de distribución [3].

Este fenómeno es un efecto que resulta de la evolución en el tiempo de las inestabilidades, puesto que habrá un momento en el que éstas llevarán el sistema a un estado metaestable. Por lo tanto, la ITS y la IBP presentarán AL como consecuencia de la disipación de energía que irá experimentando el sistema.

4.6. Campos magnéticos aplicados

Un caso extra que se va a tratar en este trabajo es la acción de un campo magnético externo, uniforme y unidireccional sobre el plasma. Del electromagnetismo se sabe que cuando una partícula se encuentra confinada bajo la acción de un campo eléctrico $\vec{E}$ y un campo magnético uniforme ($\vec{B} = B_0$) se genera un rotación por efecto del último y una traslación debida al primero. La figura 4.5 muestra la trayectoria que efectúa una partícula bajo las condiciones anteriores.

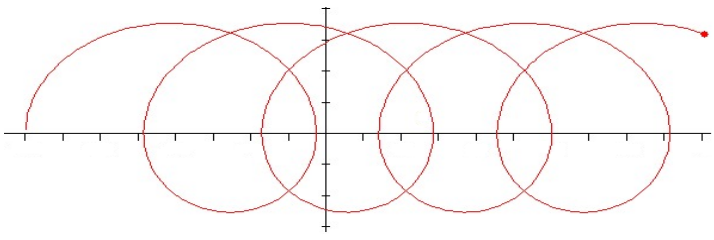


Figura 4.5: Trayectoria de una partícula al estar afectada por un campo eléctrico y un campo magnético uniforme.

Estos efectos en el movimiento de las partículas se describen en la fuerza de Lorentz completa:

$$\frac{d\vec{v_p}}{dt} = \frac{q}{m_e}\left(\vec{E_p} + \vec{v_p} \times \vec{B_p}\right) \tag{4.5}$$

El campo eléctrico se calculará de la manera usual, el campo magnético externo tendrá una magnitud predeterminada y será siempre perpendicular al campo eléctrico; la rotación que genera el campo magnético en cada partícula del sistema será calculada usando el Método de Boris.

Capítulo 5

Modelo computacional

La implementación del modelo electrostático comprendió la construcción de un código que se llamó `Plasma Computacional Grupo Plasma, Láser y Aplicaciones (PC-GPLA.py)` en Python, usando el paquete de librerías ANACONDA, que requirió una discretización de las ecuaciones involucradas y una normalización de éstas para ser incluidas dentro del ciclo computacional. La construcción de este partió de los códigos espic.py de Gibbons [15], el 1D PIC code de [16], y el Electrostatic 1D Particle in Cell in Matlab/Octave de Markidis [17].

5.1. Discretización de las ecuaciones de movimiento y campo

5.1.1. Integrador *Leap-Frog*

Es un método ampliamente utilizado para la integración de las ecuaciones de movimiento, y es útil para solucionar ecuaciones diferenciales lineales ordinarias de segundo orden. Consiste en definir la velocidad a mitad de los intervalos de integración, mientras que la posición se define en intervalos enteros como es usual [4]. En la figura 5.1 se presenta una ilustración gráfica del esquema del *Leap-Frog* (LF).

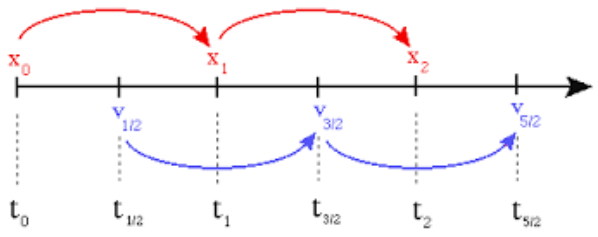


Figura 5.1: Esquema de integración del método LF [4].

De esta manera, el método LF aplicado a las ecuaciones de movimiento para el modelo electrostático se escriben de la siguiente manera:

$$x_i^{n+1} = x_i^n + v_i^{n+\frac{1}{2}} \Delta t \tag{5.1}$$

$$v_i^{n+\frac{1}{2}} = v_i^{n-\frac{1}{2}} + \frac{q_i}{m_i} E_i^n \Delta t \tag{5.2}$$

Una de las ventajas de usar este método de integración, además de su simplicidad, es que tiene la propiedad de ser reversible en el tiempo debido a la simetría con la que es definido. Lo anterior asegura que se conserve la energía y cualquier otra cantidad conservativa [4].

El esquema aquí expuesto se encuentra en la subrutina `movimientoparticulas`, contenida en el `PC-GPLA.py`, y el primer paso a medio intervalo se hace antes del ciclo principal.

5.1.2. Método del trapecio

Para solucionar la ecuación 4.1, se usó el *método del trapecio acumulativo*. Este consiste en aproximar el área bajo la curva por medio del área de trapecios definidos por intervalos. La figura 5.2 muestra la estructura del método considerando una aproximación de cuatro trapecios respecto a la curva presentada.

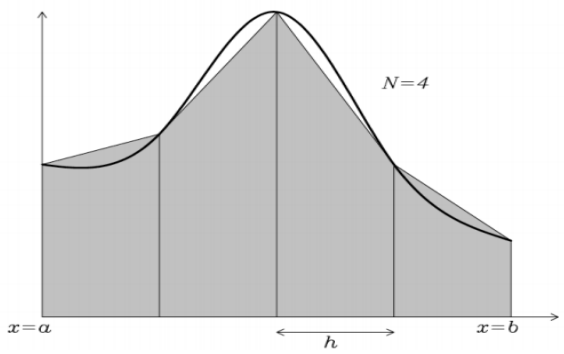


Figura 5.2: Esquema gráfico del método del trapecio acumulativo [5].

La ecuación que describe el método es la que se muestra a continuación:

$$T_h[f;a,b] = \frac{h}{2}\left(f(a) - f(b)\right) + h\sum_{j=0}^{N} f(a+jh) \tag{5.3}$$

El método del trapecio se implementó por medio de la función `cumtrapz` de la librería `Scipy`, y se encuentra en la subrutina `campo` dentro del `PC-GPLA.py` (ver 7.5). Se escogió la regla del trapecio por su facilidad de implementación y porque entregó resultados satisfactorios.

5.1.3. Método de Boris

Este método fue el utilizado para calcular el término $v \times B$ de la fuerza de Lorentz 4.5. El método consiste en dividir la rotación por etapas. Para ello, se divide la fuerza de Lorentz en sus partes eléctrica y magnética. La ecuación 5.4 muestra la fuerza reescrita para obtener una expresión que permita la implementación de la rotación.

$$\frac{\vec{v}^{n+1/2} - \vec{v}^{n-1/2}}{\Delta t} = \frac{q}{m}\left[\vec{E} + \frac{\vec{v}^{n+1/2} - \vec{v}^{n-1/2}}{2} \times \vec{B}\right] \tag{5.4}$$

Boris se percató de que el campo eléctrico podía ser eliminado definiendo las siguientes ecuaciones:

$$\vec{v}^{n-1/2} = \vec{v}^{-} - \frac{q\vec{E}}{m}\frac{\Delta t}{2} \tag{5.5}$$

$$\vec{v}^{n+1/2} = \vec{v}^{+} + \frac{q\vec{E}}{m}\frac{\Delta t}{2} \tag{5.6}$$

Al reemplazar 5.5 y 5.6 en la ecuación 5.4 se obtiene la siguiente ecuación:

$$\frac{\vec{v}^{+} - \vec{v}^{-}}{\Delta t} = \frac{q}{m}\frac{(\vec{v}^{+} - \vec{v}^{-})}{2} \times \vec{B} \tag{5.7}$$

Lo siguiente que Boris hizo fue, a través de relaciones geométricas entre los vectores de velocidad y campo magnético, (ver Figure 4-4a in [2], p. 62), deducir una expresión para realizar la rotación. El primer paso es encontrar el vector bisector a través del ángulo formado por la velocidad antes y después de la rotación. Dicho ángulo, al que la velocidad rotará en el paso de tiempo dado es: $\tan(\theta/2) = (-qB)m\Delta t/2$. El vector bisector ($\vec{v'}$) es:

$$\vec{v'} = \vec{v^{-}} + \vec{v^{-}} \times \vec{t} \tag{5.8}$$

Donde $\vec{t} = -\hat{\vec{b}}\tan(\theta/2)$. El vector $\vec{v'}$ es perpendicular al campo magnético ($\vec{t}$) y al vector de v^{-} a v^{+}, la velocidad luego de la rotación que se está buscando:

$$\vec{v^{+}} = \vec{v^{-}} + \vec{v'} \times \vec{s} \tag{5.9}$$

Donde $\vec{s} = 2\vec{t}/1 + t^2$. Este vector s es una versión del vector de rotación t escalado para satisfacer el requerimiento que la magnitud de la velocidad permanece constante durante la rotación, mostrando además que sólo se ve afectada por el campo magnético.

En el código `PC-GPLA.py`, al momento de implementar este algoritmo descrito por Boris, primero se obtiene $\vec{v^{-}}$ a través de 5.5. Luego, se efectúa la rotación mediante 5.8 y 5.9. Para finalizar, se obtiene 5.6 [18]. Se agrega que, al implementar lo anterior se construye un código 1.5D2V (1.5 dimensiones con dos componentes de velocidad).

5.2. Normalización

La normalización de unidades es importante en la simulación de fenómenos físicos, por un lado, porque permite ahorrar recursos computacionales, dado que se asegura que las cantidades y constantes, que serán procesadas por el código no acarrearán una gran cantidad de números decimales [19]. Por otro lado, normalizar las cantidades físicas involucradas posibilita que el código sea transparente, de ésta manera las ecuaciones que se solucionan son de tipo adimensional [15].

Entonces, las cantidades simuladas en el código `PC-GPLA.py` tales como tiempo, dimensión espacial, velocidad, campo eléctrico y densidad de partículas obedecen el siguiente esquema:

$$t \to \omega_p t \quad x \to \omega_p x/c \quad v \to v/c \tag{5.10}$$

$$E \to eE/m\omega_p c \qquad n_{e,i} \to n_{e,i}/n_0 \tag{5.11}$$

De acuerdo a las ecuaciones (5.10) y (5.11), las siguientes cantidades se consideraron unitarias: la carga q_e, la masa m_e, y la densidad electrónica de referencia n_0.

5.3. Condiciones de frontera

Las condiciones de frontera son un requisito indispensable en la solución de ecuaciones diferenciales, puesto que éstas determinan el marco sobre el cual el problema solucionado está actuando. Para el caso aquí tratado, se consideran condiciones de frontera periódicas, que en el momento de ser implementadas en el `PC-GPLA.py`, evitarán la creación de nuevas partículas, o en su defecto que se extravíen las ya simuladas. En el problema electrostático, dado que es unidimensional, se consideran condiciones de frontera periódicas en x. Cuando se incluye el campo magnético externo, se consideran fronteras periódicas en x e y.

5.4. Método Particle-In-Cell

El método de Particle-In- Cell es usado comúnmente para simular plasmas, gases enrarecidos, dinámica molecular de gases y otros procesos físicos del medio continuo. El método representa el fluido como una colección de supertpartículas (SP) simuladas. Las SP se mueven en el dominio descrito por dos mallas computacionales superpuestas, que se usan para calcular interacciones autoinducidas entre partículas [20].

5.4.1. Ciclo computacional

En la Figura 5.3 se muestra la estructura que tiene el ciclo implementado.

El algoritmo del PIC, para el caso de plasma electrostático, comprende las siguientes etapas que se ejecutan de acuerdo a los pasos temporales que se hayan ingresado a la simulación:

1. La solución de la ecuación de movimiento (4.2) para cada partícula; en este paso se actualiza la malla móvil.

2. En el siguiente paso se asignan las partículas a la malla fija a través de la cual se mueven, esto con el fin de calcular la densidad de carga del plasma.

3. Con la densidad de carga (ρ_x) se procede al cálculo del campo eléctrico (E_x) en la malla fija, mediante la solución de la ecuación 4.1.

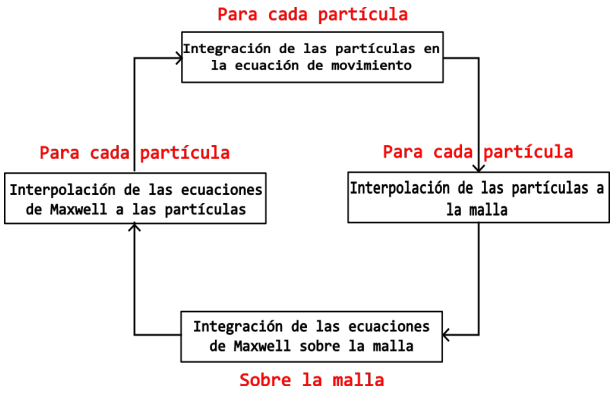


Figura 5.3: Diagrama de flujo del método PIC.

4. El campo eléctrico (E_x) calculado se asigna a las partículas y causa el movimiento de las mismas. Si el sistema simulado tiene campo magnético, en este paso se efectúa además la rotación de las partículas mediante el algoritmo de Boris (ver 5.1.3). Una vez ocurre lo anterior, se vuelve al paso 1.

Cabe anotar que, para el paso 3, en otros códigos encontrados en la literatura, se suele calcular el campo mediante la ley de Poisson, pero al hacerlo directamente con la ley de Gauss se evita un paso y se reduce el ruido computacional derivado de un cálculo adicional. Además, al solucionar directamente la ley de Gauss se asegura que se conserve la carga del sistema [21].

5.4.2. Configuración inicial

La configuración inicial consiste en la creación de las SP que modelan el plasma, así como del espacio sobre el cual éstas se van a mover. De ésta manera, se crean dos mallas: una móvil y una fija. En ésta fase del algoritmo, se consideran las condiciones presentadas en la ecuación 5.3, tal como lo propone Gibbon en su código [15].

Antes de que el ciclo computacional se efectúe, es pertinente fijar los criterios para la selección de los parámetros que le dan forma al sistema computacional, que para éste caso son el paso temporal (Δt) y el paso espacial (Δx). Así, la elección de los parámetros de la simulación se hizo cumpliendo con las condiciones expuestas por Forslund [22]:

$$\omega_p \Delta t < 2 \tag{5.12}$$

$$\lambda_d \geq \frac{\Delta x}{2} \tag{5.13}$$

Para 5.12, $\omega_p = 1$ debido a la normalización. Para 5.13, λ_d es la longitud de Debye; de acuerdo al esquema del sistema, esta cantidad representa la longitud de una celda dentro de la malla. Δt y Δx se escogen de acuerdo a la longitud de Debye y la frecuencia del plasma respectivamente, ya que como se explicó en la sección 3, estos son los parámetros que determinan las dimensiones espaciales y temporales del plasma, así se asegura que la simulación tenga sentido físico.

Además, éstos dos parámetros no son independientes entre sí. Lindman [23] dedujo que hay una correlación entre los parámetros 5.12 y 5.13, que está dada por la siguiente ecuación:

$$\delta t \leq \frac{2\pi}{\omega_p + \frac{\pi v_{th}}{\delta x}} \tag{5.14}$$

Donde, para el caso del `PC-GPLA.py` $v_{th} = 1$. La importancia de ésta correlación de variables se evidencia en el hecho de la reducción del ruido computacional, así se asegura la disminución de fenómenos no físicos.

Finalmente, la configuración inicial del sistema termina cuando se calcula la primera iteración para la densidad, y obtenida ésta, se calcula la primera iteración para el campo.

5.4.3. Malla euleriana

Es una malla que está fija con el sistema de coordenadas [24]. En el caso unidimensional, se define con puntos igualmente espaciados entre ellos por una longitud Δx definida como la longitud total de la malla dividida entre los puntos de malla requeridos. La distancia $x_{i+1} - x_i$ es igual a Δx y este espacio entre puntos constituye una celda. En cada celda de la malla Euleriana se encuentran y calculan cantidades como la densidad de carga y los campos electromagnéticos.

5.4.4. Malla lagrangiana

Es una malla móvil, o en otras palabras, está fija a las partículas del sistema [24]. En consecuencia, la malla Lagrangiana cambia conforme lo hagan las posiciones y velocidades de las SP.

5.4.5. Métodos de ponderación

Con el fin de establecer una relación entre las cantidades que se calculan en la malla lagrangiana con las de la malla euleriana, se debe implementar un método que permite esta comunicación. Existen métodos de ponderación de distintos órdenes que consideran la forma efectiva de las partículas de distintas maneras, tal como se aprecia en la figura 5.4.

El método utilizado fue el CIC, en el cual el peso de cada partícula es asociado a dos puntos de la malla y la forma efectiva es triangular [1]. La densidad, escrita de acuerdo a este esquema, es la siguiente:

$$\rho_i = \sum_p \frac{q_p}{\Delta x} W\left(x_i - x_p\right) \tag{5.15}$$

Donde x_p corresponde a la posición de las partículas, x_i a la posición de los puntos en la malla euleriana, y $W\left(x_i - x_p\right)$ corresponde a la función de ponderación. Para las cantidades de la malla móvil, es conveniente usar el mismo esquema de ponderación cada que se pase de una malla a otra. El campo eléctrico de acuerdo al CIC es:

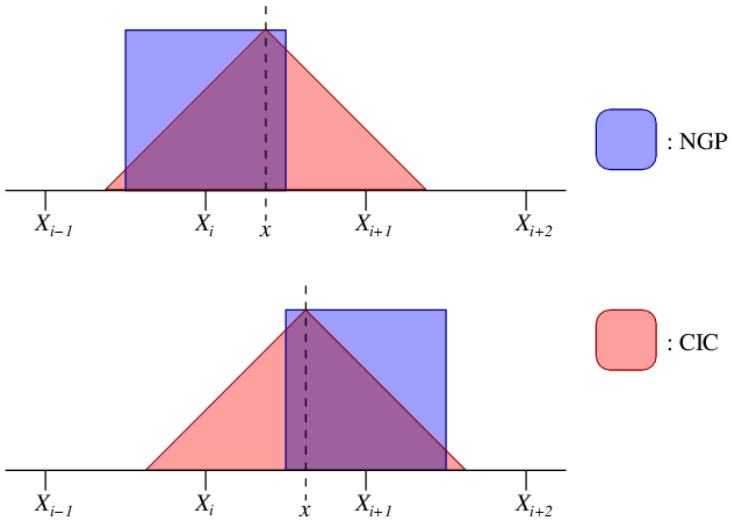


Figura 5.4: Forma efectiva de las partículas para el *Nearest Grid Point* (NGP) o esquema de orden 0, y para el *Cloud-In-Cell* (CIC) o esquema de orden 1.

$$E_i = \sum_p \frac{q_p}{\Delta x} W\left(x_i - x_p\right) \tag{5.16}$$

El método CIC, a diferencia de un método de orden superior, no es costoso computacionalmente y genera menos ruido que un método de orden cero (NGP) [19].

Parte III

Implementación y simulación

Capítulo 6

Metodología

El presente proyecto de grado se clasifica dentro de la modalidad de investigación básica dirigida, en el cual se realiza una simulación de plasma usando el paquete de librerías ANACONDA. Esto se logra mediante la implementación del ciclo computacional PIC. Por tanto, para el alcance del objetivo general se propone la siguiente metodología compuesta por las siguientes etapas:

En la primera etapa se simula el plasma electrostático. Para ello, se utiliza el paquete de librerías ANACONDA. Para la simulación de dicho plasma se usa el método de *Particle-In-Cell* (PIC), el cual exige la solución de las ecuaciones de movimiento de las partículas involucradas (electrones y iones), y la ponderación de campos ($\vec{E}$ y $\vec{B}$) y densidades (carga y masa). La forma de las ecuaciones de movimiento, y las de campos y densidades dependen del tipo de plasma que se esté estudiando, que de acuerdo al problema planteado corresponde a la solución de las ecuaciones de Poisson y la fuerza de Lorentz.

La segunda etapa corresponde a la corroboración del código construido que simula el plasma electrostático. Para esto, se parte de uno de los fenómenos mayormente tratados en la literatura: oscilaciones de plasma frío [6]. Para lo anterior, se ingresan condiciones iniciales que perturben el sistema de manera sinusoidal. Por tanto, se grafican los campos y la energía del sistema para observar el comportamiento oscilatorio característico de este fenómeno, así como la relación de dispersión ($\vec{\omega}$ como función de $\vec{k}$) para verificar la coincidencia de los datos obtenidos a partir de la simulación con la curva teórica.

Para la tercera etapa se simula la ITS. Para ello, se considera una distribución Maxwelliana que permite el modelamiento de dos flujos de electrones idénticos en densidad y con velocidades iguales pero en sentido contrario. Aquí también se perturba el sistema a través de la velocidad, que se asigna de manera aleatoria a las partículas, con la misma forma que en las oscilaciones de plasma frío. En ésta fase también se obtienen gráficas de verificación, sin embargo se considera también el espacio de fase, esto con la intención de observar la vorticidad característica de este tipo de inestabilidad [7].

En la cuarta etapa, se simula la IBP [2], esta se obtiene considerando la misma configuración de la ITS, pero para ella los flujos son de distintas densidades.

En la quinta etapa se simula el AL. Este se origina como consecuencia de la ITS o de la IBP, que no es nada más que la disipación de la energía de las partículas. Tal como en las etapas previas, se obtienen gráficas para corroborar el fenómeno. Por una parte, el espacio de fase debe mostrar partículas atrapadas en medio de los flujos, y por otra, la

energía del campo eléctrico debe presentar el amortiguamiento [2].

En la sexta etapa se incluye un campo magnético uniforme y unidireccional (B_z) a las oscilaciones de plasma frío mediante el algoritmo de Boris [18].

En la etapa final, se concluye a partir de las gráficas obtenidas a lo largo de los distintos fenómenos, corroborando mediante ellos el comportamiento propio del plasma electrostático: su capacidad de generar campos.

Finalmente, a partir de los resultados obtenidos, se pretende publicar artículos concernientes a la investigación realizada, para así interactuar con la comunidad científica de la física del plasma en congresos y revistas afines.

Capítulo 7

Desarrollo del PC-GPLA.py

El `PC-GPLA.py`, código que permite simular plasma electrostático y plasma bajo la acción de un campo magnético, es una implementación secuencial, construida a través de subrutinas. Se hace la salvedad que se construyó en `Python 2.7`.

Primero que todo, es pertinente mencionar las variables declaradas, que son esenciales para la posterior implementación del código:

- Número de SP: `nparticulas`
- Puntos de la malla: `npuntos_malla`
- Longitud de la malla computacional: `longitud_malla`
- Paso espacial: `dx`
- Paso temporal: `dt`
- Número de pasos temporales: `npasos_temporales`
- Velocidad media del haz: `vh`
- Amplitud de perturbación de la posición: `aP`
- Amplitud de perturbación de la velocidad: `vP`
- Magnitud del campo magnético: `Bo`

Además de las declaraciones previas, se asignan los espacios en memoria para cantidades como:

- Posiciones en `x`,`y`, `z`
- Velocidades `v`,`vx` y `vy`
- Densidades: electrónica `rhoe`, iónica `rhoi` y total `rhot`
- Campo eléctrico `Ex`

– Energías: cinética (`EnergiaK`), potencial (`EnergiaP`), y total (`EnergiaT`)

Luego de repasar la declaración de las variables importantes para el diseño del `PC-GPLA.py`, a continuación se presentan cada una de las subrutinas, explicando la manera como fueron diseñadas y su funcionamiento.

7.1. Subrutina `cargarposicion`

En esta subrutina se cargan las SP al sistema; a su vez, se calculan las dimensiones de la malla computacional y el espacio entre SP. La carga y la masa se guardan en variables computacionales pero tienen valor igual a uno, debido a la normalización expuesta en 5.2. En `cargarposicion` se implementa un ciclo que asegura que las partículas se distribuyan a lo largo de toda la malla fija (ver 5.4.3), y se agrega una perturbación sinusoidal, mediante `aP`, a las posiciones de las SP.

7.2. Subrutina `cargarvelocidad`

Esta subrutina asigna la velocidad que tendrá inicialmente cada SP. La velocidad que se asigne dependerá del fenómeno que quiera ser simulado. Esto se logra a través de la variable `tdistribucion`, que de acuerdo al valor entrará a condicionales diseñados con instrucciones que permitan el acceso a distribuciones de velocidad específicas (ver A). Las posibilidades son:

– `tdistribucion = 0` para simular plasma frío, lo que permitirá la visualización de oscilaciones (ver 4.2).

– `tdistribucion = 1` configurar el sistema computacional para simular ITS (ver 4.3).

– `tdistribucion = 2` aquí se ajusta el sistema computacional para obtener SP bajo la acción de campos eléctrico y magnético (ver 4.6). Se aclara que se debe tener un valor para el campo $\vec{B}$, esto a través de la varible `Bo`.

– `tdistribucion = 3` con ésta opción se obtiene la IBP. Las velocidades de las SP se distribuyen en dos distribuciones de Maxwell, una con velocidad media cercana a 0 y densidad mayor a la otra, que a su vez tendrá una velocidad media igual a `vh` (ver 4.4).

7.3. Subrutina `particula_cf`

En esta subrutina se asegura que las SP permanezcan dentro de la malla computacional aplicando las condiciones de frontera expuestas en 5.3. Esta función recibe como argumento a `longitud_malla`, que será guardada en una variable local denominada `xtamanio`.

7.4. Subrutina `densidad`

Recibe como argumento la carga que tendrá cada una de las SP, que se consideró como unitaria en los fenómenos simulados. Asimismo, se definen dos variables locales `j1` y `j2` que corresponden a los índices de la malla Lagrangiana (ver 5.4.4) y se definen como enteras. Aquí se calcula la densidad de los electrones por medio del esquema CIC (ver 5.4.5), que se implementa en el ciclo dentro de esta función. Se usan condiciones de frontera periódicas para asegurarse que las SP no se salgan de la malla fija.

7.5. Subrutina `campo`

A la densidad (`rhoe`) ya calculada se le agrega la densidad de los iones (`rhoi`), que se considera neutral. La densidad total (`rhot`) generada se integra de acuerdo a la regla del trapecio (ver 5.1.2). Para este fin se uso la función `cumtrapz` de la librería `Scipy` de ANACONDA. Lo anterior corresponde a integrar la ecuación 4.1 para obtener el campo eléctrico (`Ex`). Puesto que se genera una onda correspondiente al campo, que a su vez va a estar almacenado en una matriz proporcional a los puntos de malla, se le resta un término de offset (`edc`). De nuevo, se usan condiciones de frontera periódicas.

7.6. Subrutina `movimientoparticulas`

En ésta sección del código se actualizan las posiciones de las SP por medio de la expresión 4.2, que ha sido escrita en diferencias, usando el método del LF (ver 5.1.1). El campo calculado en el paso anterior se guarda en cada paso del ciclo acá implementado en la variable `exi`, mediante el método CIC. Dicha variable corresponde a la magnitud que tiene el campo en una celda de la malla, y es la que entra en la fuerza de Lorentz. Además, en ésta rutina, se considera si hay o no campo magnético (`Bo`). Si el campo $\vec{B}$ está ausente, como se describió en la sección 4.1, se ejecuta lo descrito de manera previa. En caso contrario, se efectúa lo anterior más la rotación de velocidades presentada en la sección 5.1.3.

7.7. Subrutina `diagnosticos`

En ésta función se generan las gráficas de campo, densidad, espacio de fase y funciones de distribución. Tales gráficas se guardan de acuerdo a una frecuencia calculada a partir del paso de tiempo `dt` determinado por el usuario. Al final de la subrutina se calculan las energías cinética, potencial y total del sistema.

7.8. Subrutina `historial`

Se grafica la energía del sistema contra el tiempo, una vez termina el ciclo principal. De igual modo como en la subrutina `diagnosticos`, se guarda la gráfica y los datos generados por la ejecución de la simulación.

7.9. Subrutina `relacion_dispersion`

Se calculan y se grafican las relaciones de dispersión del campo eléctrico para las oscilaciones de plasma frío y la ITS (ver C). Esta función recibe como argumentos la variable `tdistribution`, así el usuario determina la relación de dispersión que desea mapear. Dentro de ésta subrutina, se hace un cálculo de las frecuencias angulares mínima y máxima, así como del número de onda mínimo y máximo, tal como lo propone el código `KEMPO.m` de Omura [25]. Las frecuencias espacial y temporal del campo se obtienen usando la transformada de Fourier por medio del algoritmo FFT (ver B). Para graficar el resultado anterior se convocan funciones como `meshgrid` del paquete `Scipy`, y `contourf` del paquete `Pylab`. Finalmente, los archivos generados se guardan en carpetas especificadas por el código, tal como se explica en la sección 7.10.

7.10. Subrutina `gestion_archivos`

En la subrutina `diagnósticos` se generan una serie de archivos de tipo imagen (extensión PNG), éstos se guardan a través de la función `gestion_archivos` en carpetas una vez se ha corrido el código. Para que ésta funcione, primero es necesario hacer unas declaraciones previas para reservar espacio en memoria:

- `Archivos_Densidades`
- `Archivos_Campo`
- `Archivos_Efase`
- `Archivos_Fdistribucion`
- `Archivos_Trayectorias`

Para el caso de las energías y las relaciones de dispersión, no es necesaria dicha declaración puesto que se obtiene una sola gráfica de cada una por cada corrida del código. A su vez, en la sección inicial del `PC-GPLA.py` se encuentran declarados dos módulos necesarios para la manipulación de archivos:

- `os` es un módulo para manipulación de archivos.
- `sys` módulo para copiar, mover y renombrar archivos.

7.11. Subrutina `nextpow2`

Ésta subrutina se encarga de aproximar un número a la potencia de dos más cercana. Es necesaria para la calibración del eje de frecuencias, puesto que sin ella las relaciones de dispersión estarían corridas respecto al cero, y sin ello la FFT no funcionaría correctamente.

7.12. Programa principal

En esta parte se encuentra la configuración inicial del sistema que involucra las subrutinas `cargarposicion` y `cargarvelocidad`. Además, se lleva a cabo el primer paso del LF como se propone en la sección 5.4.2, se calcula la densidad y campo y se grafican las cantidades físicas propuestas en la función `diagnostico` para el instante de tiempo $t = 0$.

Aquí también está el ciclo principal (PIC), en dónde se llaman las subrutinas descritas anteriormente en el siguiente orden:

1. `movimientoparticulas`
2. `particula_cf`
3. `densidad`
4. `campo`

El orden en que se llaman las subrutinas corresponden con el algoritmo descrito en el apartado 5.4. También dentro del ciclo principal se grafican las cantidades de la función `diagnóstico` para el análisis de la evolución temporal de las mismas. De manera adicional, se guardan los valores de campo eléctrico en todos los instantes de tiempo para, una vez terminado el ciclo principal, calcular su transformada de Fourier y obtener su relación de dispersión (ver C) en la subrutina con el mismo nombre. Finalmente, se llama la subrutina `historial`, y se grafican la energía del sistema simulado, así como la trayectoria de una partícula bajo la acción de un campo en el caso que `Bo` sea diferente de cero.

Parte IV

Discusión y resultados

Capítulo 8

Discusión y resultados

En ésta sección se presentan los resultados obtenidos a partir del código `PC-GPLA.py` construido en Python, usando el paquete de librerías ANACONDA.

8.1. Oscilaciones de plasma frío

Para obtener los resultados correspondientes a oscilaciones de plasma frío, la configuración se tomó de Birdsall y Langdon [2]. Ésta fue de 2048 partículas, 256 puntos de malla, 150 pasos temporales, una longitud de malla de 4π, un $dt = 0{,}1$, y una perturbación sinusoidal en posición de amplitud 0,001.

La figura 8.1 corresponde a una configuración de partículas que debido a su movimiento fueron capaces de generar, a través de cambios en la densidad, un campo eléctrico oscilatorio, mostrando así una de las características clave del plasma: campos autogenerados (ver 4.1). A su vez, la figura 8.2 presenta un intercambio entre la energía cinética y potencial, como en un oscilador armónico simple (ver 4.2), y una oscilación en la energía total posiblemente por efecto de la perturbación.

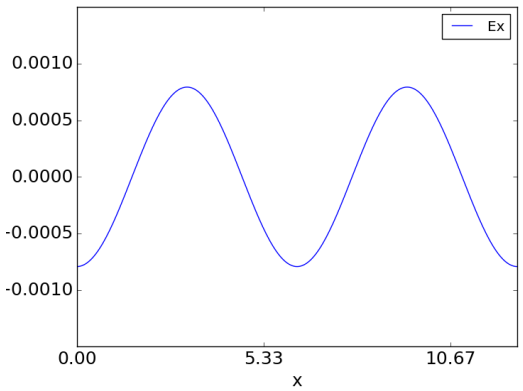


Figura 8.1: Campo eléctrico en $t = 150$ para oscilaciones de plasma frío.

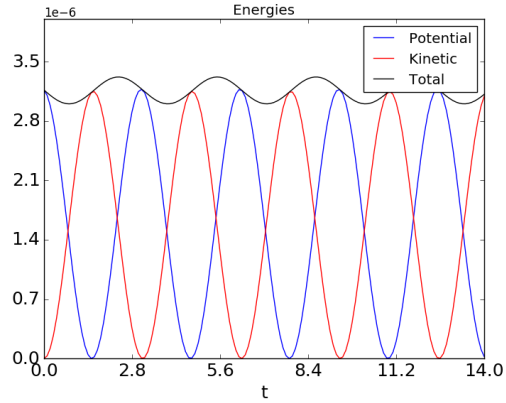


Figura 8.2: Energías para oscilaciones de plasma frío.

La figura 8.3 es la relación de dispersión para el fenómeno en cuestión y muestra las

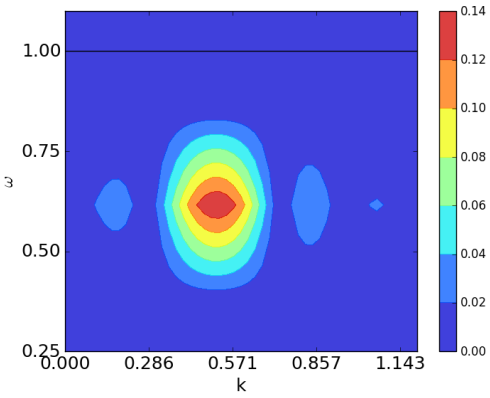


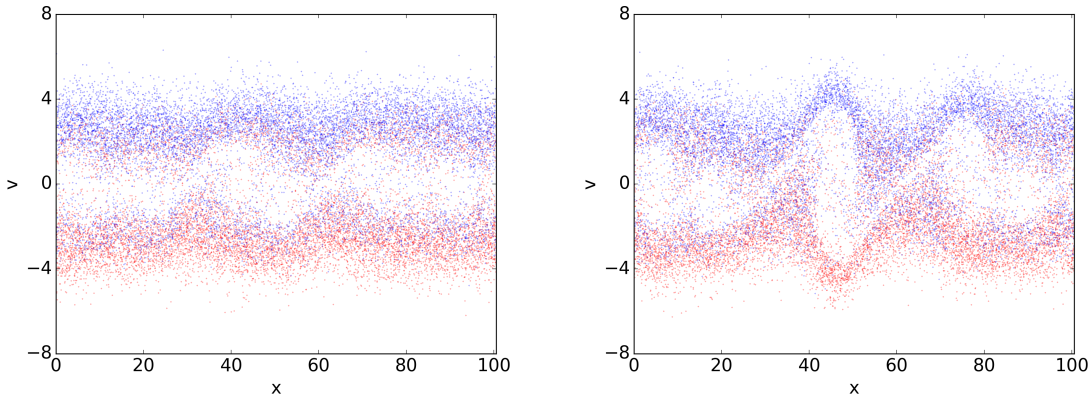
Figura 8.3: Relación de dispersión para oscilaciones de plasma frío: teórica (línea negra constante) y simulada(diagrama de colores).

frecuencias espaciales y temporales una contra la otra (ver Anexo C). La barra vertical muestra las intensidades del campo eléctrico, que fueron obtenidas al calcular la FFT sobre una matriz espacio temporal. En la gráfica se observan diferentes colores, éstos corresponden a tendencias del sistema a oscilar a distintas frecuencias. Así, colores cálidos como el rojo, el amarillo y el naranja muestran una mayor tendencia a oscilar a dichas frecuencias, mientras que, colores fríos como los verdes en diferentes tonalidades, o azules claros, todo lo contrario a lo anterior. Entonces, se observa que el sistema en cuestión oscila en $k \approx 0{,}571$, ya que la intensidad del campo eléctrico es mayor en dicha región, y con valores distintos de ω por debajo de ω_p tal como lo expone [6].

8.2. Inestabilidad *Two-Stream*

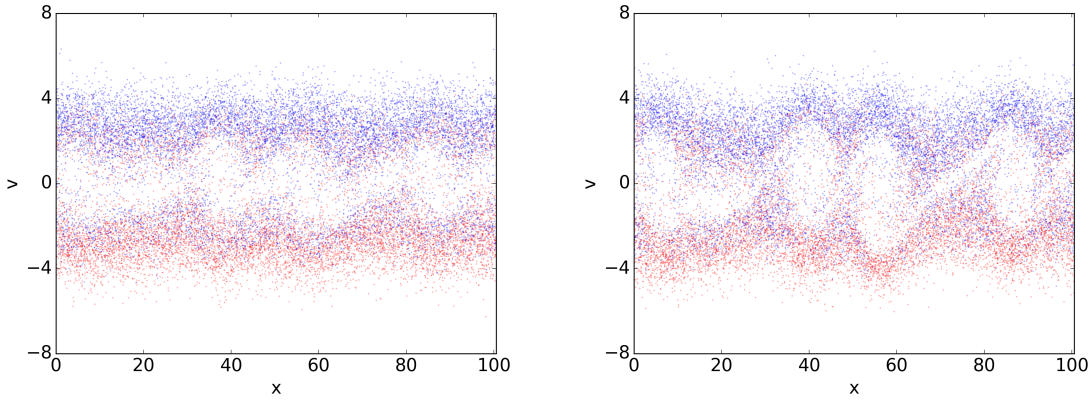
La ITS comprendió la configuración de un sistema con 20000 partículas, 1024 puntos de malla, una velocidad media de los haces de 2,5, una longitud de malla de 32π y 150 pasos temporales. Para obtener resultados se perturbó el sistema de manera sinusoidal a través de la amplitud de la velocidad con tres distintos valores 0,0, 0,1, y 0,5. Esto se hizo con el fin de determinar la incidencia de la perturbación en el desarrollo de la inestabilidad y su posterior amortiguamiento.

Se obtuvieron gráficas de la relación de dispersión y gráficas de espacio de fase en dos instantes de tiempo para cada amplitud de perturbación. Los instantes de tiempo seleccionados para graficar el espacio de fase de la ITS fueron dos: el momento en que se empieza a desarrollar la inestabilidad, al que se denominó *etapa de transición*, y el último paso temporal. Las figuras 8.4, 8.5 y 8.6 son los espacios de fase con diferentes amplitudes de perturbación. Estos resultados corresponden a dos flujos de electrones idénticos en un plasma unidimensional, tal como los obtuvo Fitzpatrick [16]. Además, en las subfiguras 8.4(b), 8.5(b) y 8.6(b) se muestra la vorticidad propia de la ITS [7], que se genera debido a la aceleración de un flujo de electrones contra otro de igual densidad, tal como se explica en la sección 4.3. Respecto al tiempo de aparición de la inestabilidad se agrega que éste cambia. En la figura 8.4(a), la ITS aparece como efecto del movimiento de las partículas, indicando que el sistema se conduce a un estado de metaestabilidad, además de mostrar que las partículas experimentan un calentamiento [1]. Las figuras 8.5(a) y 8.6(a), presentan para la etapa de transición elegida un desarrollo mayor respecto a 8.4(a), mostrando así



(a) t=100 (b) t=150

Figura 8.4: Espacio de fase para ITS en plasma frío: sin perturbación.

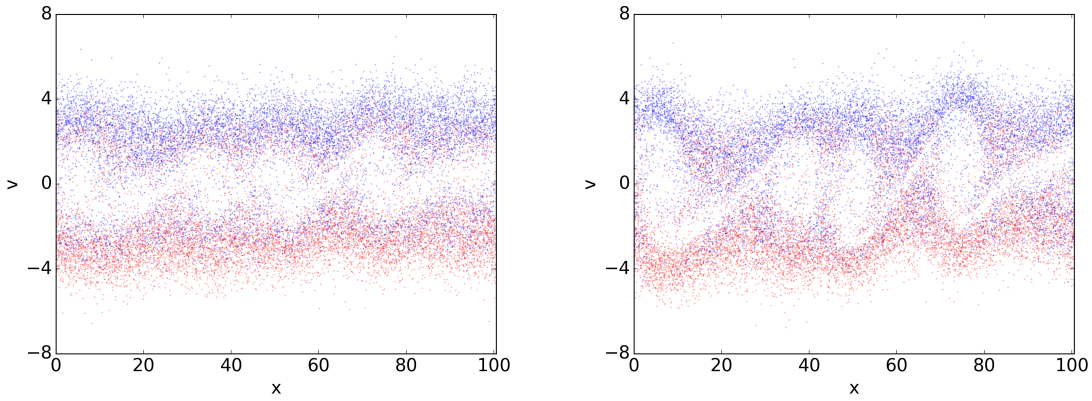


(a) t = 100 (b) t = 150

Figura 8.5: Espacio de fase para ITS en plasma frío: amplitud de la perturbación de 0,1.

que a mayor perturbación más rápido aparece la ITS.

Respecto a las relaciones de dispersión 8.7, 8.8 y 8.9, se presentan los resultados simulados a través del `PC-GPLA.py` junto con las curvas teóricas correspondientes. La línea negra constante es la frecuencia ω_p y la curva negra corresponde a la ecuación C.11 para la ITS. Para los valores simulados, se usó la misma escala de colores que para el plasma frío, teniendo en cuenta que para obtenerlos se recurrió a la FFT (ver Anexo C), tal como se explica en la sección 7.9. En las tres gráficas se observa que todas las partículas oscilan por debajo de la frecuencia del plasma y siguiendo la curva teórica con una mayor concentración al principio de la misma para amplitud de perturbación igual a cero. Sin embargo, a medida que crece la perturbación salen de los primeros números de onda de la relación de dispersión teórica y se acercan a los últimos. Esto se puede ver como una transición de los datos correspondientes a los números de onda, ya que parten de un número de onda entre 0 y 0,14, como se observa en la figura 8.7, y se concentran entre 0,42 y 0,56, tal como se ve en la figura 8.9. Además, se evidencia un incremento en las intensidades de ésta gráfica



(a) t = 100 (b) t = 150

Figura 8.6: Espacio de fase para ITS en plasma frío: amplitud de la perturbación de 0,5.

respecto a las otras, ya que las magnitudes llegan hasta 40, mientras que para 8.7 y 8.8 las magnitudes alcanzan 18 y 16 respectivamente.

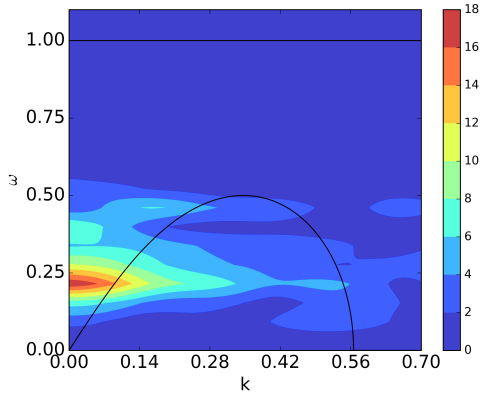


Figura 8.7: Relación de dispersión para la ITS en plasma frío: teórica y simulada sin perturbación.

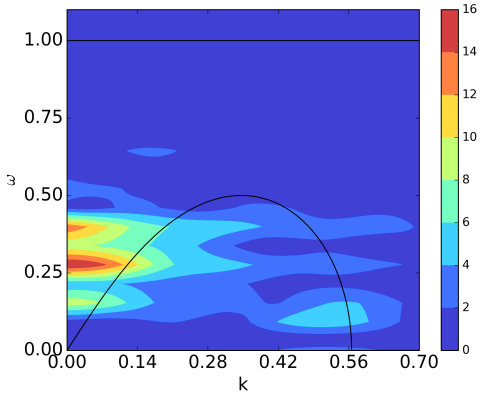


Figura 8.8: Relación de dispersión para la ITS en plasma frío: teórica y simulada con perturbación de 0,1 de amplitud.

8.3. Inestabilidad *Beam-Plasma*

La obtención de resultados para la IBP comprendió una configuración de 16384 SP, 128 puntos de malla, una longitud de malla de 16π, 300 pasos temporales y sin perturbación. Se consideró que el flujo con mayor velocidad tenía el 10 % del total de las SP, ésto con la intención de mantener los requerimientos para que se generara ésta inestabilidad [2] y una velocidad media de $vh = 4$. El flujo cercano al reposo contiene el resto de partículas del sistema, y su velocidad fue calculada de acuerdo a lo propuesto por Tigik,Ziebell, Yoon, y Kontar [13].

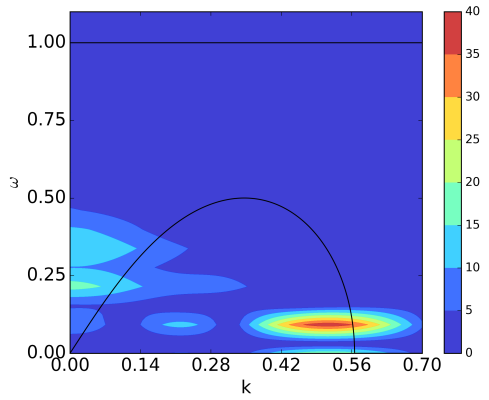
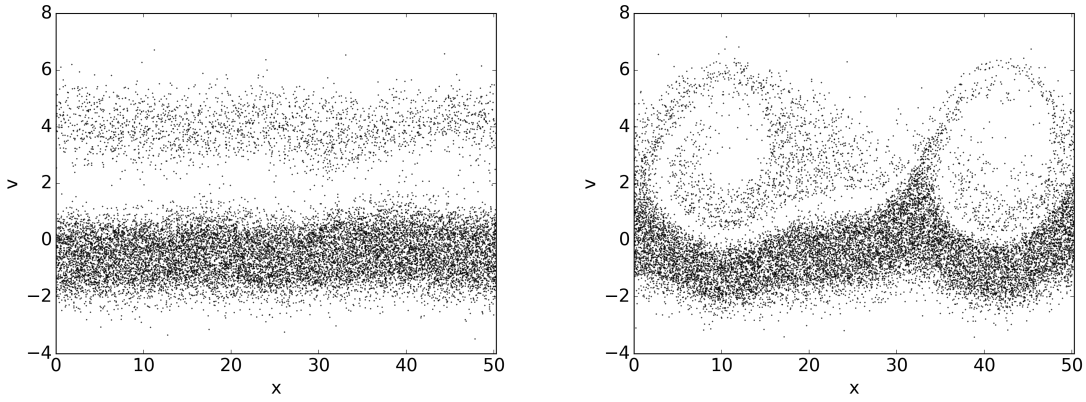


Figura 8.9: Relación de dispersión para la ITS en plasma frío: teórica y simulada con perturbación de 0,5 de amplitud.

Las figuras 8.10(a) y 8.10(b) muestran el espacio de fase en la etapa de transición y la etapa final, que se generan debido a que uno de los flujos (el de menor densidad), se inyecta a mayor velocidad que el plasma, mostrando así los vórtices que pueden ser observados en la subfigura 8.10(b). Se nota que los resultados concuerdan con los expuestos por Fox, Williams y Messina [26] a partir de un código paralelizado. Cabe agregar que los resultados aquí expuestos son satisfactorios, puesto que fueron obtenidos mediante un código secuencial.



(a) t=118 (b) t=256

Figura 8.10: Espacio de fase para IBP en plasma frío: sin perturbación.

8.4. Amortiguamiento de Landau

El AL se obtiene al ejecutar las configuraciones de la ITS o de la IBP durante más pasos temporales. Para las pruebas efectuadas con el `PC-GPLA.py` se le ingresaron a la simulación 1200 pasos. El amortiguamiento se visualiza en el espacio de fase, donde se observa el "trapping" de las partículas que sugieren Birdsall y Langdon [2]. Los vórtices se caracterizan por ser cerrados, indicando que pocas partículas aún conservan la energía proveniente de

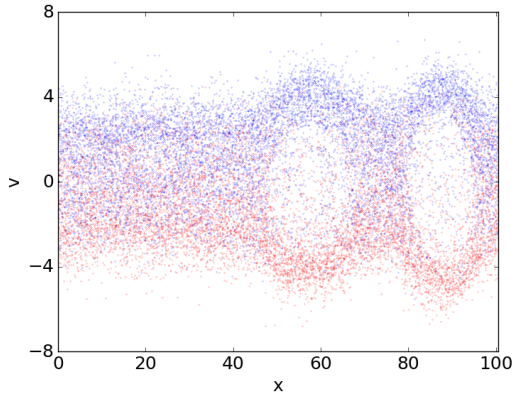


Figura 8.11: Amortiguamiento de Landau en el espacio de fase para la ITS

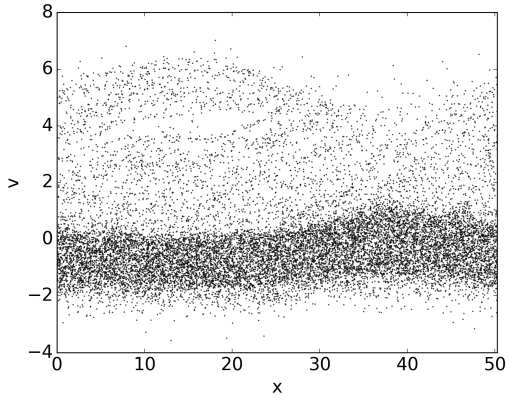


Figura 8.12: Amortiguamiento de Landau en el espacio de fase para la IBP.

la perturbación, lo que indica un estado de metaestabilidad del sistema. Evidencia de ésto se observa en las figuras 8.11 y 8.12.

Asimismo, el AL se evidencia en las gráficas de energía del campo eléctrico. En la figura 8.13 se muestra este efecto para la ITS, evidenciando que la caída de la energía no es tan abrupta. Por su parte, la figura 8.14 muestra el mismo efecto pero de manera más pronunciada, mostrando así la relación que hay entre ésta disipación de energía y el "trapping" presente en las figuras 8.11 y 8.12.

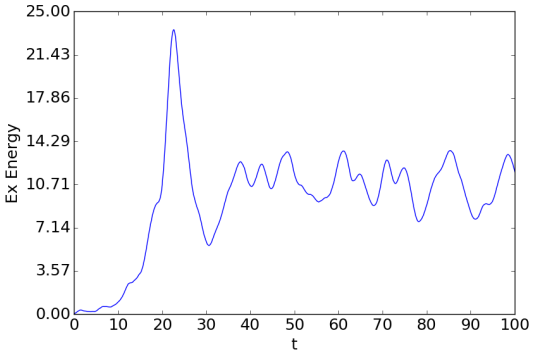


Figura 8.13: Amortiguamiento de Landau visto a través de la energía del campo eléctrico para la ITS

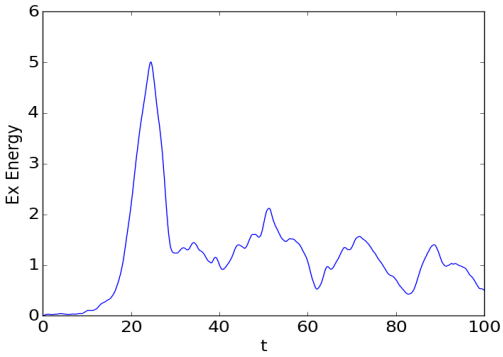


Figura 8.14: Amortiguamiento de Landau visto a través de la energía del campo eléctrico para la IBP.

8.5. Campo magnético uniforme aplicado a un flujo de electrones

Se aplicó un campo magnético en la dirección perpendicular (z) a un flujo de electrones con 10000 SP, una malla computacional de longitud 10 y 100 puntos de malla. Los electrones

del flujo seguían una distribución de Maxwell con velocidad media igual a $v_h = 8$.Se aclara que la configuración en esta parte del código no es tan importante como en los otros fenómenos, ya que el interés era observar la rotación producida por el campo magnético y comprobar que el método de Boris estuviera bien implementado. La magnitud del campo magnético deber ser lo suficientemente grande para que la fuerza magnética equipare la eléctrica y se observe la rotación que este genera (ver 4.6) dentro del espacio simulado. Los resultados se obtuvieron con $Bo = 5$. En la figura 8.15 se muestra el efecto de un campo magnético sobre tres SPs, que no es nada más que una rotación debida a la sola presencia del campo magnético.

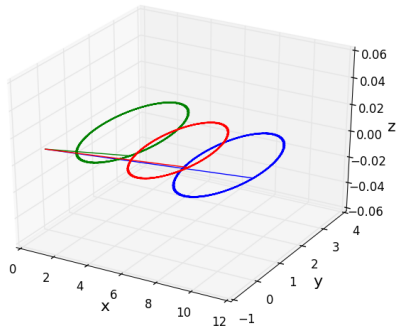


Figura 8.15: Trayectoria de SPs afectadas por un campo magnético externo uniforme ($Bo = 5$)

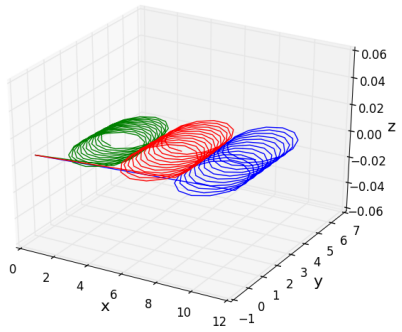


Figura 8.16: Trayectorias SPs distintas bajo la acción de campos eléctrico y magnético ($Bo = 5$) perpendiculares entre sí.

La figura 8.16 presenta las trayectorias para varias SPs, esta vez considerando el campo eléctrico que producen éstas. Las trayectorias describen la rotación y traslación de las SPs al efectuar la espiral esperada por la teoría (ver 4.6).

Parte V

Conclusiones y Trabajo Futuro

Capítulo 9

Conclusiones y trabajo futuro

9.1. Conclusiones

Se construyó un código en Python que permite simular oscilaciones de plasma frío, inestabilidades Two-Stream y Beam-Plasma, y amortiguamiento de Landau en plasma electrostático. Además, se implementó un algoritmo para la aplicación de un campo magnético uniforme y unidireccional. También, se comprobó que Python, mediante su paquete de librerías científicas ANACONDA, es una herramienta computacional que por su fácil lectura e implementación, permite un desarrollo de simulaciones aproximadas a la realidad, y a su vez logra ser didáctica al momento de entender el PIC y los métodos numéricos que éste involucra. Así, los resultados obtenidos para los fenómenos estudiados están de acuerdo a los conseguidos por distintos autores: las gráficas para plasma frío corroboran los resultados de Martin [1]; los espacios de fase y las relaciones de dispersión para las distintas pruebas para la ITS concuerdan con los de Fitzpatrick [16] y Martin [1]; para la IBP, los espacios de fase se asemejan a los resultados de Fox, Williams y Messina [26]; y para el AL, los espacios de fase obtenidos a partir de la ITS y la IBP, así como las respectivas gráficas de energía, entran en concordancia con las expuestas por Birdsall y Langdon [2]. Los fenómenos acá estudiados son representativos y básicos dentro de la simulación del plasma y los algoritmos usados para simularlos pueden ser modificados con facilidad para estudiar métodos numéricos distintos a los usados o para a partir de estos, estudiar fenómenos más complejos. Así, este trabajo sienta las bases para una línea de investigación en plasma computacional dentro del GPLA.

9.2. Artículos publicados

J.S. Blandón, J.P. Grisales, H. Riascos, *Electrostatic plasma simulation by Particle-In-Cell method using ANACONDA package*, (proceso de publicación en la IOP Proceedings).

Fue presentado en calidad de ponencia oral en el *V Congreso Nacional de Ingeniería Física* llevado a cabo en Medellín del 26 al 30 de septiembre del presente año.

9.3. Trabajo futuro

El trabajo futuro, respecto a la parte teórica y algorítmica, consiste en acercar el modelo computacional a la realidad mediante la consideración de: campos magnéticos generados por el movimiento de los electrones, colisiones entre partículas, extender a un modelo bidimensional (2D) y tridimensional (3D).

Respecto a la parte computacional se propone paralelizar el código para acelerar su ejecución y aplicar métodos de convergencia para hacerlo más eficiente computacionalmente.

Parte VI

Anexos

Apéndice A

Código Plasma Computacional Grupo Plasma Láser y Aplicaciones

El siguiente enlace está direccionado a un repositorio en `GitHub` que contiene el código desarrollado: https://github.com/jsblandon/PC-GPLA/

La figura A.1 presenta el diagrama de flujo del código desarrollado. Para un mayor entendimiento de las variables involucradas ver la sección 7.

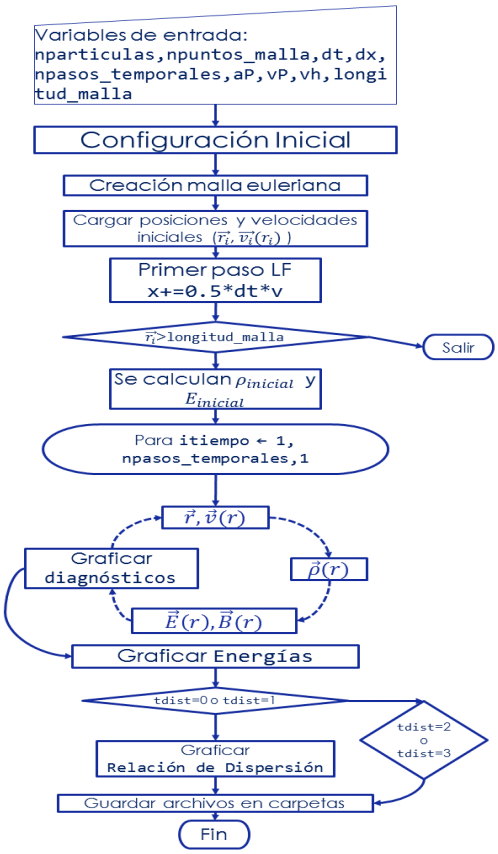


Figura A.1: Diagrama de flujo del `PC-GPLA.py`.

Apéndice B

Transformada Rápida de Fourier

La FFT es un algoritmo para calcular la Transformada Discreta de Fourier (DFT), ya sea en tiempo o en frecuencia. Para efectos de este trabajo, se explica la segunda. La FFT es un algoritmo que está basado en el principio fundamental de descomponer el cálculo de la DFT de una secuencia de longitud N, en varias DFT más pequeñas y sucesivas [27].

Se considera el problema de calcular la serie compleja de Fourier:

$$X(j) = \sum_{k=0}^{N-1} A(k) \cdot W^{jk}, \qquad j = 0, 1, ..., N-1 \tag{B.1}$$

Donde $A(k)$ son los coeficientes complejos de Fourier y W está definido como:

$$W = e^{2\pi/iN} \tag{B.2}$$

El cálculo de B.1 requeriría N^2 operaciones, lo que significa una multiplicación compleja seguida por una suma de la misma naturaleza. Debido a ésto, Cooley y Tukey dedujeron un algoritmo que requería un almacenamiento de máquina menor al que se efectuaría al calcular la DFT tal como estaba definida. Este fue llamado el *algoritmo de Cooley-Tukey*, que más tarde se conocería como FFT. La ventaja de este se encuentra en el hecho que se requieren menos de $2N \log_2 N$ operaciones, haciendo el algoritmo efectivo al momento de calcular la transformada de Fourier computacionalmente. Para la derivación del algoritmo se recomienda referirse a [28]. En la figura B.1 se presenta el esquema de operaciones del algoritmo mencionado.

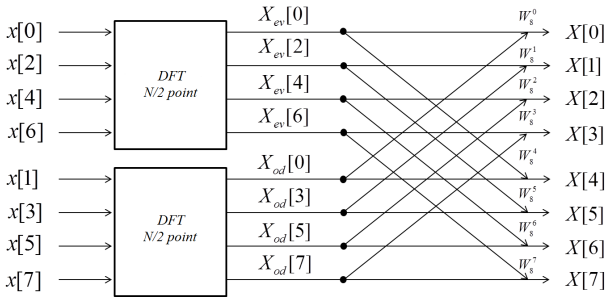


Figura B.1: Esquema del cálculo del algoritmo de Cooley-Tukey o FFT.

Apéndice C

Relaciones de dispersión

Una relación de dispersión asocia la frecuencia angular con el número de onda y describe si una onda sufre o no distorsión en la forma en que se propaga, y de que manera lo hace.

C.1. Relación de dispersión de oscilaciones de plasma frío

Para oscilaciones de plasma frío se consideró únicamente el movimiento de los electrones, los cuales oscilan alrededor de sus posiciones de equilibrio como lo haría un oscilador armónico simple cuya frecuencia de oscilación es la frecuencia del plasma ω_p (ver 3.2). Lo anterior se puede comprobar partiendo del siguiente conjunto de ecuaciones como lo exponen Birdsall y Langdon [2]:

$$m\frac{d\vec{v_p}}{dt} = q\vec{E_p} \tag{C.1}$$

$$\nabla \cdot \rho\vec{v} + \frac{\partial \rho}{\partial t} = 0, J = \rho\vec{v} \tag{C.2}$$

$$\nabla \times H = J + \epsilon_0 \frac{\partial \vec{E}}{\partial t} \tag{C.3}$$

La dependencia entre posición y tiempo se asume como $e^{i(k\cdot v_0 - \omega t)}$, donde $i = \sqrt{-1}$. Además, se considera que las cantidades serán la suma de su valor en equilibrio más una perturbación dependiente de x y t tal como lo muestra C.4.

$$F(x,t) = F_0 + F_1(x,t) \tag{C.4}$$

Donde F corresponde a las cantidades involucradas, F_0 es la cantidad en equilibrio y F_1 su perturbación.

Con estas consideraciones se tienen las siguientes ecuaciones linealizadas:

$$-mi(\omega - \vec{k}\cdot\vec{v_0})\vec{v_1} = q\vec{E_1} \tag{C.5}$$

$$-mi(\omega - \vec{k}\cdot\vec{v_0})\vec{v_1} = q\vec{E_1} \tag{C.6}$$

$$\rho_1 = \rho_0 \frac{\vec{k}\cdot\vec{v_1}}{\omega - \vec{k}\cdot\vec{v_0}} \tag{C.7}$$

Y tras un procedimiento algebraico, involucrando a C.5,C.6 y C.7, se obtiene la función dieléctrica lineal:

$$\frac{\epsilon}{\epsilon_0} = 1 - \frac{\omega_p^2}{(\omega - \vec{k}\cdot\vec{v_0})^2} \tag{C.8}$$

De donde las soluciones para ondas longitudinales se obtienen cuando $\epsilon = 0$ y son:

$$\omega = \vec{k}\cdot\vec{v_0} \pm \omega_p \tag{C.9}$$

Como la velocidad inicial es igual a 0 en el caso estudiado, la gráfica que muestra la relación de dispersión es la siguiente:

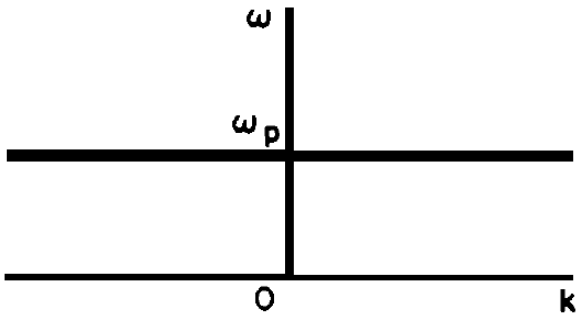


Figura C.1: Relación de dispersión para el plasma frío.

C.2. Relación de dispersión ITS

En la ITS la función dieléctrica lineal se calcula como se hizo en la sección previa, al aplicar las ecuaciones de movimiento y continuidad separadamente para dos flujos de electrones y sumándolos en la ecuación de campo. El resultado es el siguiente:

$$\frac{1}{\epsilon_0}\epsilon(\omega, k) = 1 - \frac{\omega_{p1}^2}{(\omega - \vec{k}\cdot\vec{v_{01}})^2} - \frac{\omega_{p2}^2}{(\omega - \vec{k}\cdot\vec{v_{02}})^2} \tag{C.10}$$

Las raíces para el sistema estudiado, con $\omega_{p1} = \omega_{p2}$ y $v_{01} = -v_{02}$, se encuentran cuando $\epsilon(\omega, k) = 0$ y son:

$$\omega = \sqrt{k^2 v_0^2 + \omega_p^2 \pm \omega_p \sqrt{(4k^2 v_0^2 + \omega_p^2)}} \tag{C.11}$$

La figura C.2 muestra la relación de dispersión teórica para la ITS:

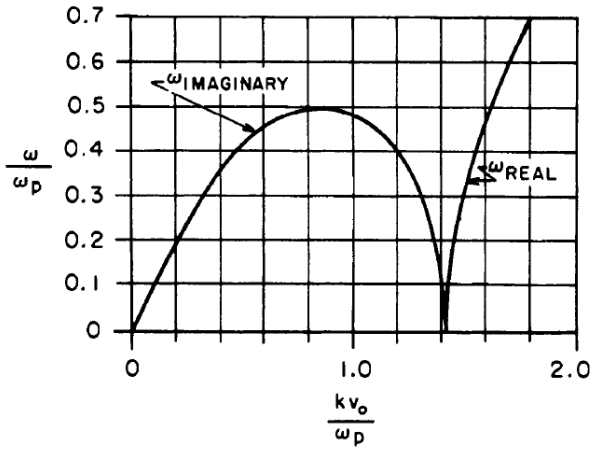


Figura C.2: Relación de dispersión para la ITS considerando uno solo de los cuadrantes [2].

Bibliografía

[1] D. Martin, "Electrostatic pic simulation of plasmas in one dimension," *Project Report, the university of Manchester (janury)*, 2007.

[2] C. K. Birdsall and A. B. Langdon, *Plasma physics via computer simulation*. McGraw-Hill, 1984.

[3] S. Markidis, "Electrostatic waves, landau damping, two-stream instability and the particle-in-cell method," 2014, [Online; accessed 2016-10-29]. [Online]. Available: https://www.pdc.kth.se/education/computational-plasma-physics/electrostatic-waves-and-particle-in-cell-method

[4] S. McMillan, "Leap frog integrator," *Computational Physics.[Online] https://www.physics.drexel.edu/students/courses/Comp_Phys/Integrators/leapfrog/*, 2013.

[5] G. Prieto, "Capítulo 3 integración numérica," [Online; accessed 2016-10-30]. [Online]. Available: http://wwwprof.uniandes.edu.co/~gprieto/classes/compufis/integracion.pdf

[6] F. Chen Francis, "Introduction to plasma physics," 1974.

[7] R. L. Morse and C. Nielson, "One-, two-, and three-dimensional numerical simulation of two-beam plasmas," *Physical Review Letters*, vol. 23, no. 19, p. 1087, 1969.

[8] C. Gonzalez, J. Arteaga, Y. Gomez, J. Osorio, J. Jaramillo, and H. Riascos, "Simulation of plume-plasma expansion with one-dimensional particle-in-cell," in *Journal of Physics: Conference Series*, vol. 370, no. 1. IOP Publishing, 2012, p. 012033.

[9] C. Analytics, "Anaconda software distribution," 2016, [Online; accessed 2016-10-06]. [Online]. Available: https://contiuum.io

[10] P. Debye and E. Hückel, "De la theorie des electrolytes. i. abaissement du point de congelation et phenomenes associes," *Physikalische Zeitschrift*, vol. 24, no. 9, pp. 185–206, 1923.

[11] R. J. Goldston and P. H. Rutherford, *Introduction to plasma physics*. CRC Press, 1995.

[12] S. Humphries Jr, "Charged particle beams," *New York, Wiley-Interscience, 1990, 849 p.*, vol. 1, 1990.

[13] S. Tigik, L. Ziebell, P. Yoon, and E. Kontar, "Two-dimensional time evolution of beam-plasma instability in the presence of binary collisions," *Astronomy & Astrophysics*, vol. 586, p. A19, 2016.

[14] L. D. Landau, "On the vibrations of the electronic plasma," *Zh. Eksp. Teor. Fiz.*, vol. 10, p. 25, 1946.

[15] P. Gibbon, "Plasma wake acceleration," 2014, [Online; accessed 2016-06-20]. [Online]. Available: https://indico.cern.ch/event/285444/contributions/1636921/

[16] R. Fitzpatrick, "Computational physics: An introductory course," *Lecture Notes. The University*, 2006.

[17] S. Markidis, "Electrostatic 1d particle in cell in matlab/octave," 2016, [Online; accessed 2016-08-16]. [Online]. Available: https://www.pdc.kth.se/education/computational-plasma-physics

[18] P. I. C. Consulting, "Particle push in magnetic field (boris method)."

[19] G. Lapenta, "Particle-in-cell method, a brief description of the pic method," *Centrum voor Plasma Astrofysica Katholieke Universiteit Leuven, unpublished*, 2006.

[20] P. I. C. Consulting, "Fundamentasl of particle-in-cell method."

[21] S. Markidis and G. Lapenta, "The energy conserving particle-in-cell method," *Journal of Computational Physics*, vol. 230, no. 18, pp. 7037–7052, 2011.

[22] D. W. Forslund, "Fundamentals of plasma simulation," *Space Science Reviews*, vol. 42, no. 1-2, pp. 3–16, 1985.

[23] E. Lindman, "Dispersion relation for computer-simulated plasmas," *Journal of Computational Physics*, vol. 5, no. 1, pp. 13–22, 1970.

[24] M. M. Woolfson and G. J. Pert, *An introduction to computer simulation*. Oxford University Press,, 1999.

[25] Y. Omura, "One-dimensional electromagnetic particle code: Kempo1."

[26] G. C. Fox, R. D. Williams, and G. C. Messina, *Parallel computing works!* Morgan Kaufmann, 2014.

[27] A. Oppenheim, R. Schafer, and J. Buck, "Discrete-time signal processing," 1999.

[28] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex fourier series," *Mathematics of computation*, vol. 19, no. 90, pp. 297–301, 1965.